Single Event Analysis and Fault Injection Techniques Targeting Complex Designs Implemented in Xilinx-Virtex Family Field Programmable Gate Array (FPGA) Devices

Melanie Berg, AS&D Inc. in support of NASA/GSFC

Melanie.D.Berg@NASA.gov

Kenneth Label: NASA/GSFC

Hak Kim: AS&D Inc. in support of NASA/GSFC

To be presented by Melanie D. Berg at the Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop, La Jolla, CA, May 19-22, 2014.

List of Acronyms

- Configuration logic block (CLB)
- Device under test (DUT)
- Edge-triggered Flip-Flop (DFF)
- Fault Injection (FI)
- Field Programmable Gate Array (FPGA)
- Linear Energy Transfer (LET)
- Lookup table (LUT)
- Number of configuration bits (NT)
- Single Event Effects (SEEs)
- Single Event Transient (SET)
- Single Event Upset (SEU)
- Static random access memory (SRAM)
- Total number of configuration bits for one fault injection campaign ($\#Bit_{FI\_inj}$)
- Time for total fault injection (T_{fl_total})
- Time to flip one configuration bit (t_{FI_inj})
- Time to wait for error response (t_{wait})
- Time to correct the inverted configuration bit (t_{corr})
- Time to reset functionality (t_{rst})

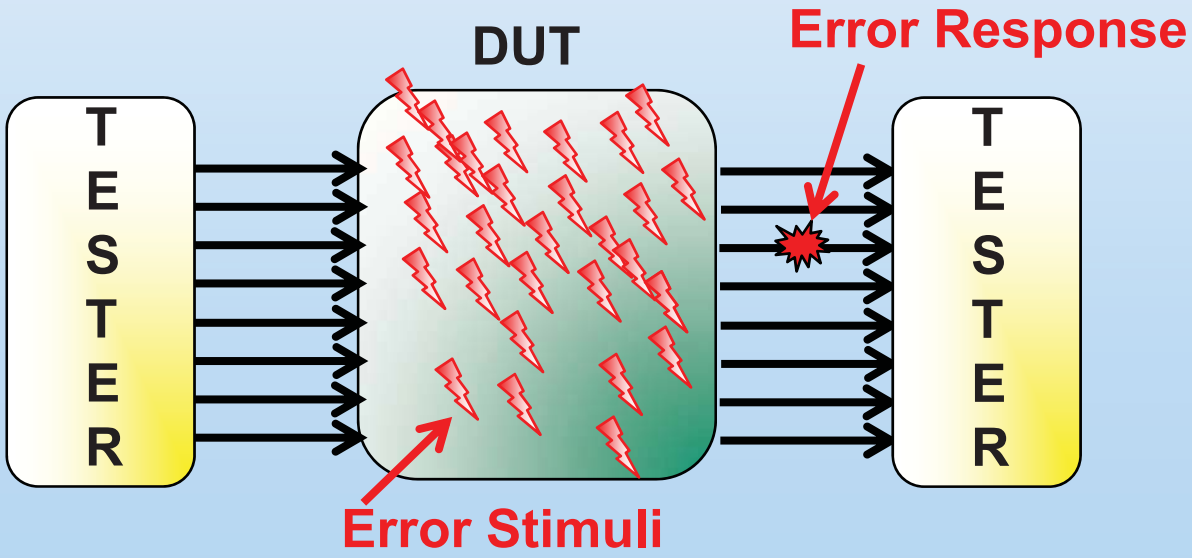
Abstract

- Informative session regarding SRAM FPGA basics.
- Presenting a framework for fault injection techniques applied to Xilinx FPGAs.
- Introducing an overlooked time component that illustrates fault injection is impractical for most real designs as a stand-alone characterization tool.
- Demonstrating procedures that benefit from fault injection error analysis.

Question: Why are you performing fault injection?…

Single Event Upset Analysis

- We define error-event stimuli as sources applied to internal DUT structures that can potentially cause DUT malfunction. For example:
  - Ionizing particles,
  - Laser pulses, or
  - Forced logic-state inversion (fault injection (FI)).

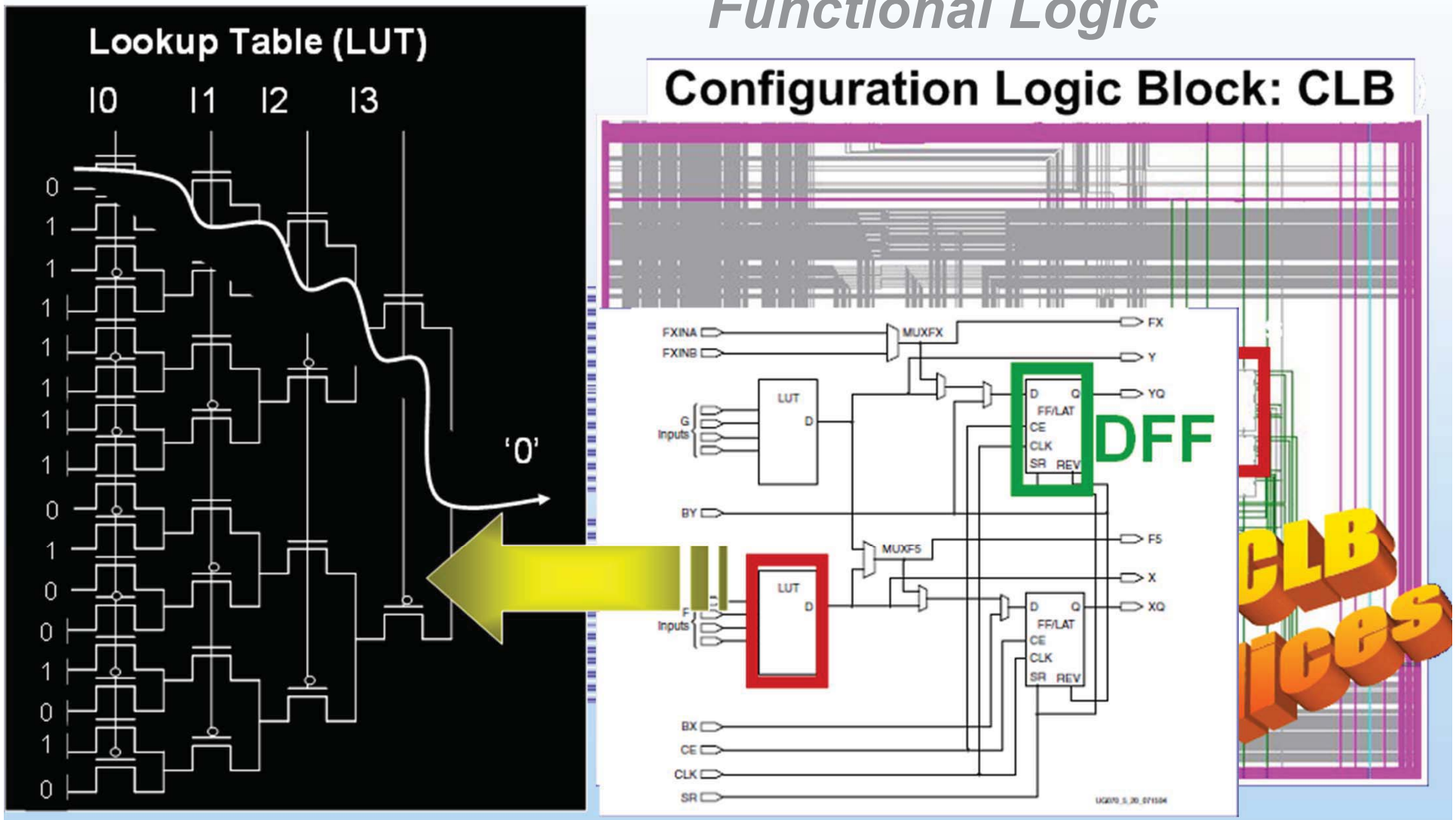


Application of Error Stimuli

- **Involves a variety of considerations such as:**
  - **Invoking a large enough event space for proper statistics,**
  - **Avoiding unrealistic fault accumulation, and**
  - **Respecting the amount of time required for an error event to reach an observable test point.**
- **In this presentation, we focus on:**
  - **The application of error-event stimuli to Xilinx Virtex FPGA devices in the form of fault injection.**

http://www.xilinx.com/

General Xilinx Virtex FPGA Architecture



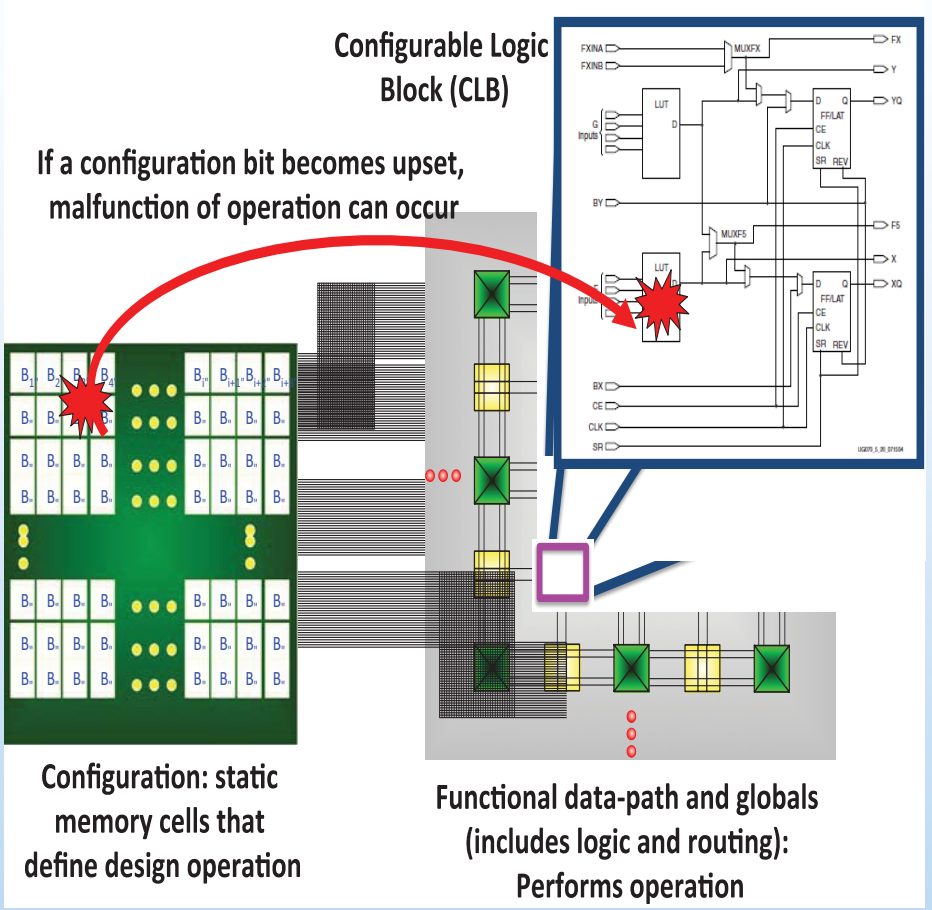
SRAM-Based FPGAs: SEUs and Fault Injection (FI)

- SRAM-based FPGAs can incur SEUs in:
  - Configuration bits,
  - Functional logic (data path transistors – combinatorial logic and flip-flops (DFFs)),
  - Global routes, or
  - Hidden logic structures (inaccessible to the user).
- Although all internal FPGA structures are susceptible to SEUs, we limit the scope of this study to fault injection in just the configuration memory.

We study how configuration-bit SEUs affect associated Xilinx components.

SRAM-Based Configuration FPGA FI

- **SRAM-based configuration fault injection:**
  - **Flip the state of a configuration bit, and**
  - **Wait and attempt to detect if an error occurs after the configuration bit-state is changed (not all configuration bits are used – not all flips will cause an error response).**



http://www.xilinx.com/

Goals of SRAM-based Configuration FPGA FI

- Determine which configuration bits can affect circuit behavior (sensitive configuration bits).
- Investigate Potential Error Responses.

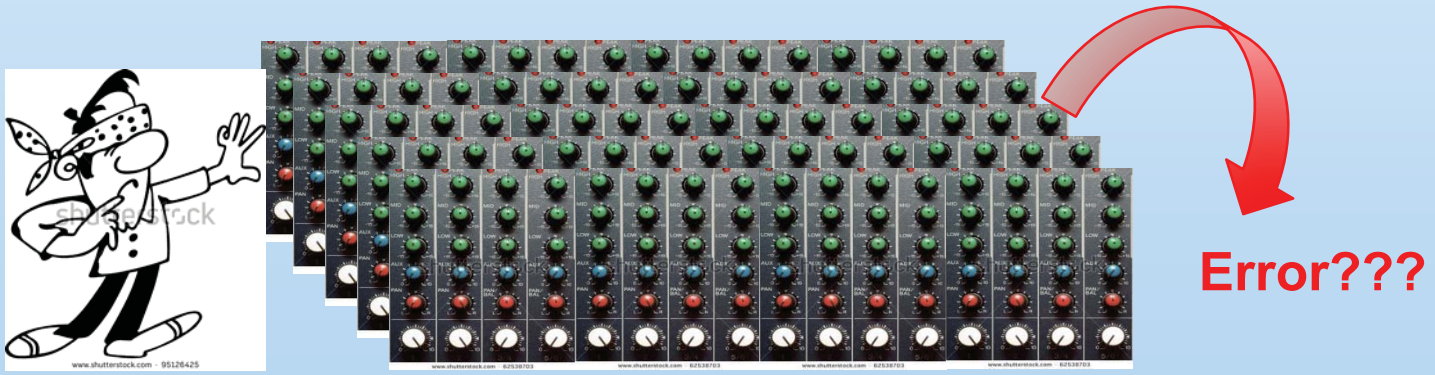
Configuration-bit FI does not determine error rates.

Determining Sensitive Configuration Bits

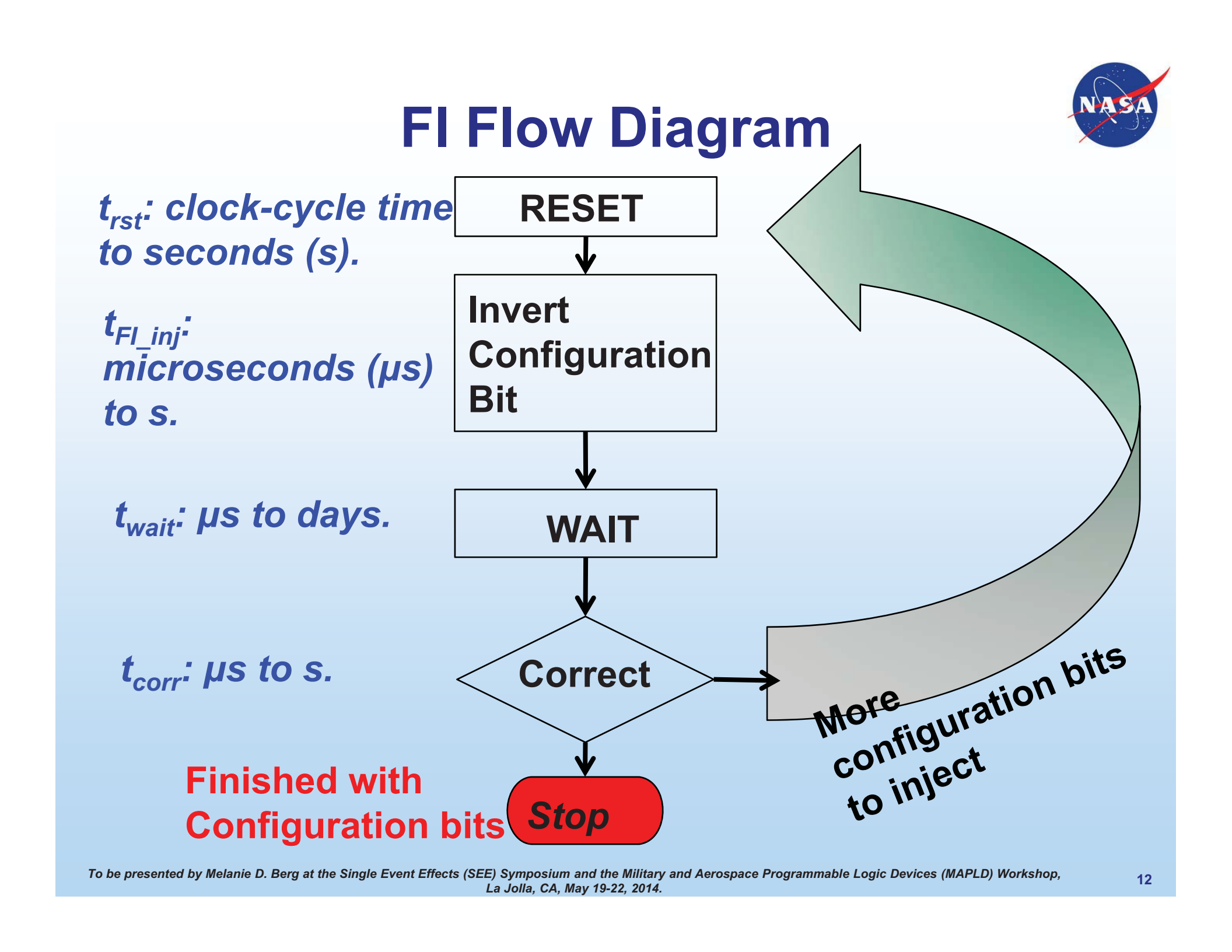
- Not all configuration bits are used by a design.
- Used and un-used configuration bits can affect a design when upset.
- Impossible to determine every bit that can affect a design because of:
  - Number of configuration bits,
  - State space complexity, and
  - Time to perform injection.

Investigating Potential Error Responses

- FI is analogous to turning a knob and waiting to see if an error occurs per knob turn:
  - Real design – the wait time after the knob is turned can be significant (complex state-space).
  - Many knobs to turn… impossible to turn every one for a real design. So must pick a subset.
- Hence, not all error responses can be observed.



FI Flow Diagram



State Space Complexity and Wait Time

- When a configuration bit is flipped:
  - The associated circuitry must be turned on (active) in order to determine if the inverted configuration bit will affect operation.
  - Depending on the state of circuitry usage during FI, error responses can differ.
- Examples of complex state space and design operation:
  - Design startup process will be different than normal operation.
  - On-off states.

FI and The Design Startup Phase of Operation

- **What happens during the beginning of operation… real-design versus test-circuit?**
  - **Real designs: built-in-self-test, computer boot-up, register loading, communication set-up/adjustments, etc…**
  - **Test circuits: usually straight forward and there is little to no special start-up sequencing, but there can be.**
- **FI during set-up will not reflect most true error responses because real operation has not begun.**

User must be aware of which states are operating when the configuration bit is inverted.

Time Required for SRAM-Based FPGA FI

- We define the total number of configuration bits for one fault injection campaign as $\#Bit_{FI\_inj}$.

$$\#Bit_{FI\_inj} < NT$$

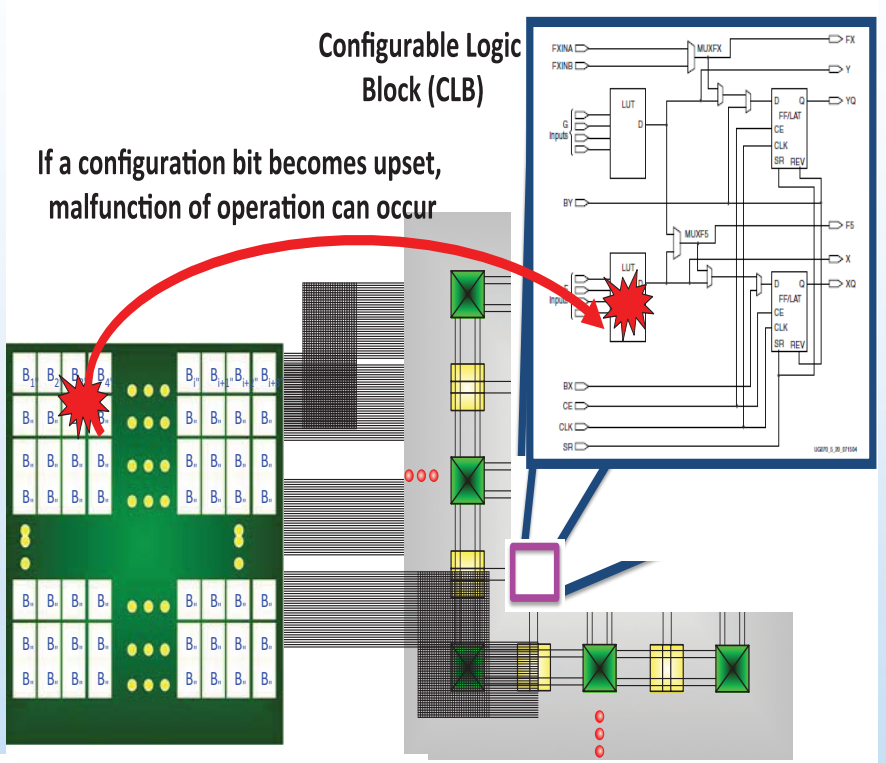
- The total time required (T_{fl_total}) for a fault injection campaign is:

$$T_{fl\_total} = \underset{millions}{\#Bit_{FI\_inj}} * (\underset{\mu s - s}{t_{FI\_inj}} + \underset{\mu s - days}{t_{wait}} + \underset{\mu s - s}{t_{corr}} + \underset{clock\ cycle - s}{t_{rst}})$$

- The fault injection tool can control t_{FI_inj} and t_{corr}.
- However t_{wait} and t_{rst} are design dependent.
- In real designs t_{wait} can be days. As an example: think about a test campaign for designs with no errors injected – can take days to find a design flaw. State space exploration takes time.

Understanding Configuration FI Error Responses

- Inverting the state of a configuration bit can have a variety of error responses:
  - Stuck state (broken route or incorrect function definition),
  - Incorrect logic behavior, or
  - Oscillations (global routes).
- Most injections will cause broken routes, such as stuck faults
- Problem with stuck faults:
  - It is not known which configuration bit controls which portion of the design.
  - If each configuration bit FI is not held long enough, error responses will not be observed.



http://www.xilinx.com/

Why Are You Performing FI?

- Do you want to know which configuration bits will affect your design if in error? (e.g., used configuration bits – or unused configuration bits that can cause contention upon SEU)
- If so, finding these "sensitive" configuration bits can be very time consuming.
  - In most real designs, it will be impossible to find every sensitive configuration bit.
  - Test designs generally have simple state spaces and can easily be fault injected.
  - However, keep in mind that error responses of test circuits will not necessarily reflect actual designs.

Example of Fault Injecting a Counter (1)

- Table is an example of a 4-bit counter:
  - "4-bit counter" refers to 4 DFFs, combinatorial logic, and a clock. The number of associated configuration bits is unknown.
  - 2^4 states = 16 states.
- With a 100 MHz clock, it would take (16 × 10 ns) = 160 ns to span the entire 4-bit counter's state space.
- 32-bit counter: 2^{32} states = 4294967296 states.
- With a 100 MHz clock, it would require approximately 43 s to span the entire 32-bit counter's state space.

4-Bit Counter: States 0..15

b3	b2	b1	b0	State
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Example of Fault Injecting a Counter (2)

- The frequency that a bit inverts its state for a counter reduces by 2 as the bit order increases:
  - Least significant bit (b_0) flips state every clock cycle
  - Next bit (b_1) flips state every two clock cycle, etc…
- Bits that flip frequently are easy to test, i.e., we can determine if they are stuck in a logic state quickly

4-Bit Counter: States 0..15

b3	b2	b1	b0	State
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Example of Fault Injecting a Counter (3)

- The most significant bit is in a static state for half of the state space traversal time.
- Hence, with a 100 MHz clock, the most significant bit of the 32-bit counter is expected to stay at a logic ‘0’ or stay at a logic ‘1’ for approximately 22.5 s.
- You will need to wait this long to determine if a configuration SEU has caused the DFF-bit to be stuck at ‘0’.

4-Bit Counter: States 0..15

b3	b2	b1	b0	State
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Time Required for 32-bit Counter Fault Injection

- The total time required (T_{fl_total}) for a fault injection campaign is:

$$T_{fl\_total} = \#Bit_{FI\_inj} \times (t_{FI\_inj} + t_{wait} + t_{corr} + t_{rst})$$

8-million *1 ms* *22.5 s* *1 ms* *10 ns*

- T_{fl_total} is roughly 2100 days (5.7 years) for fault injecting a 32-bit counter.
- 32-bit counters (and larger) are common in many designs.
- Structures such as multipliers or dividers are exponentially more complex.
- Configuration fault injection of a full design is impractical.

Benefits of FI

- Investigation of a portion of error responses (but not all possibilities can be covered).
- Evaluation of how DUT errors can affect a system, e.g., how faults can affect other devices
- Validation of test equipment (however, with the understanding that not all cases can be covered).

Conclusions (1)

- No tool is available that lists all configuration bits that can affect a design (sensitive bits).
- When performing FI, SRAM-based FPGAs can have millions of configuration bits to inject.
- The combination of the number of configuration bits and t_{wait} make FI impractical for developing a full characterization for DUT SEU error responses.
- SRAM-based configuration FI will not provide error rates. It provides a glimpse into a DUT's various error responses.
- Benefits of FI are: DUT and system level error response investigation.

Conclusions (2)
Putting Things into Perspective

- A study that suggests that it can fault inject a design in seconds or hours:
  - Is not covering the entire space of the design and will have a limited view of error responses, and/or
  - Will not be able to determine all configuration bits that can affect design operation upon an SEU, and/or
  - Is not implementing t_{wait}.
- A study that states that via SRAM-based fault injection they will provide an error rate is incorrect.
  - This type of injection cannot calculate error rates. You force a bit to flip, you know it will flip. Hence, no rate is calculated. Only error responses are observed if the FI is held long enough.

Acknowledgements

- Defense Threat Reduction Agency (DTRA)
- NASA Electronic Parts and Packaging (NEPP) Program
- Radiation Effects and Analysis Group (REAG) led by Kenneth LaBel and Jonathan Pellish.